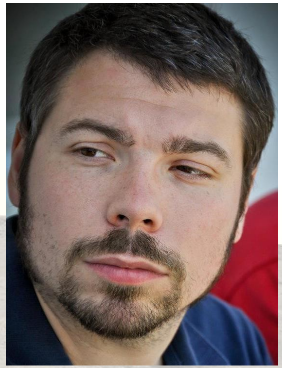
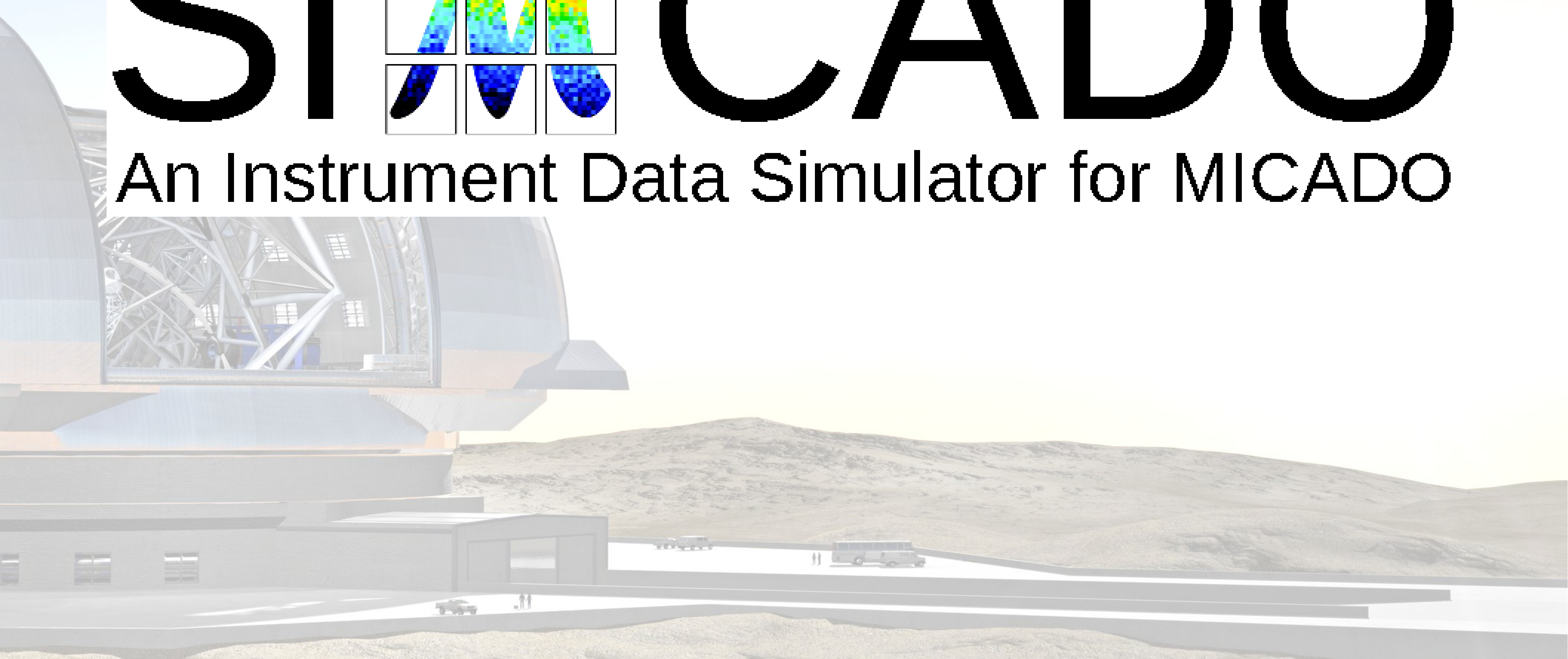
The A* Simulation team in Vienna


Kieran Leschinski and Oliver Czoske



Joao Alves, Werner Zeilinger, Rainer Köhler, Michael Mach



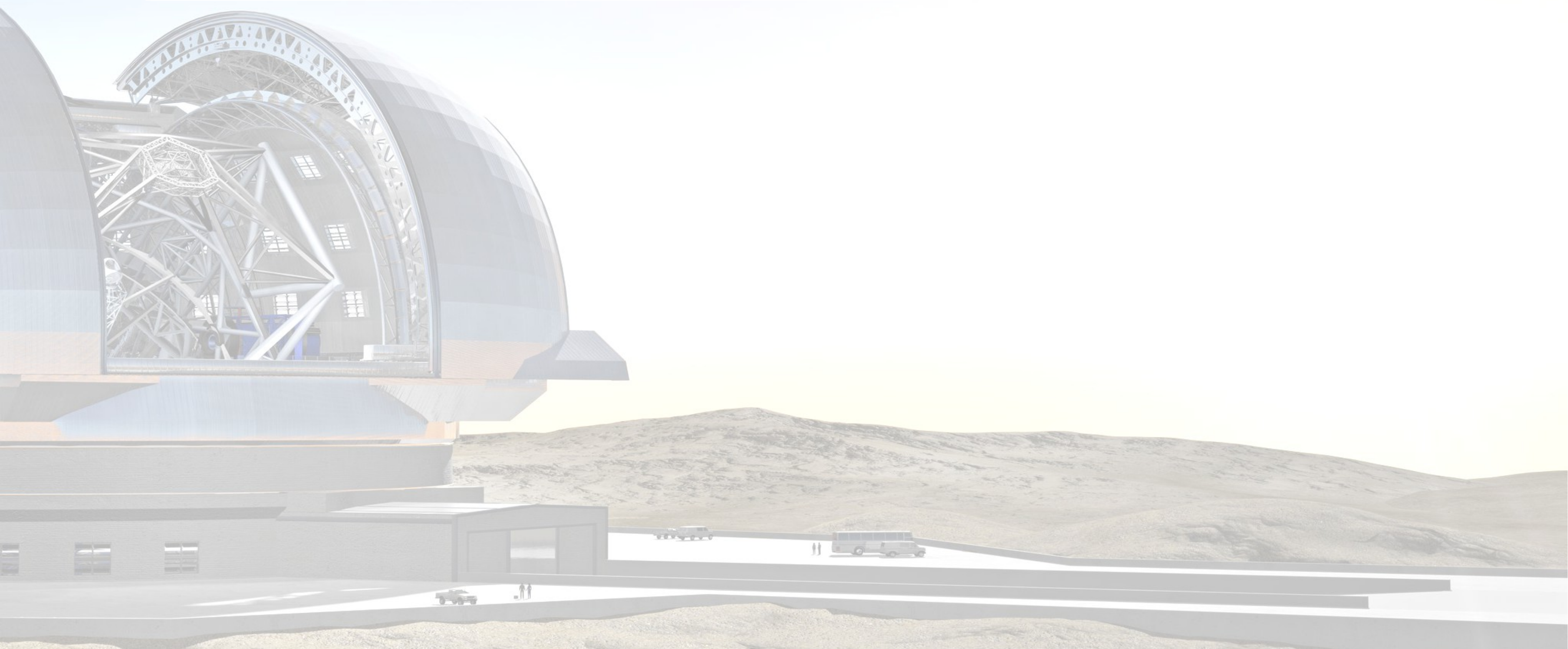
SiMCADO

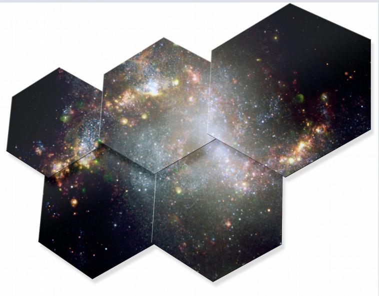
An Instrument Data Simulator for MICADO

SimCADO

An Instrument Data Simulator for MICADO

What is SimCADO?



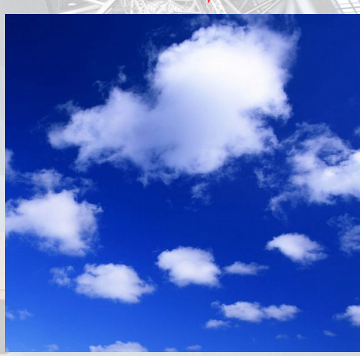


SimCADO is a **python** package which allows one to **simulate** mock **detector** array **readouts** based on the current design of MICADO

SimCADO mimics changes to the incoming source photons and produce detector-array readout files

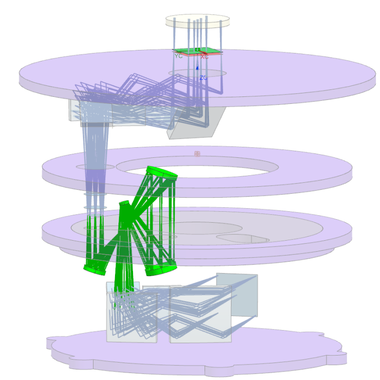
- the atmosphere
- the E-ELT
- MICADO
- the detector plane array

Source

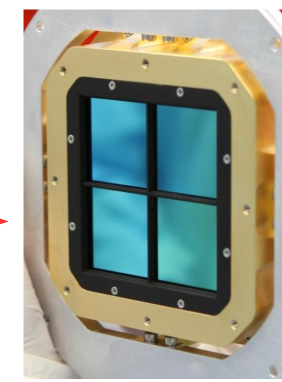


Atmosphere

Telescope



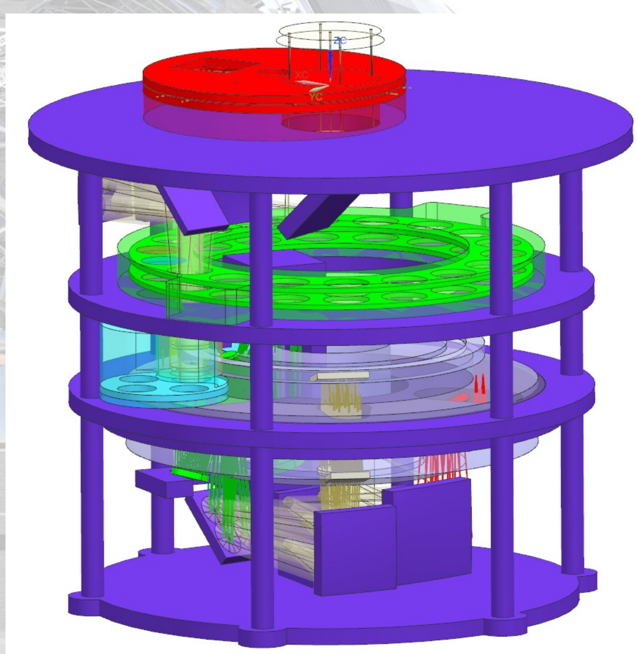
MICADO



H4RG

MICADO will be the E-ELT's NIR wide-field imager

- 0.7 – 2.4µm (IYJHK)
- Diffraction limited
- MCAO and SCAO



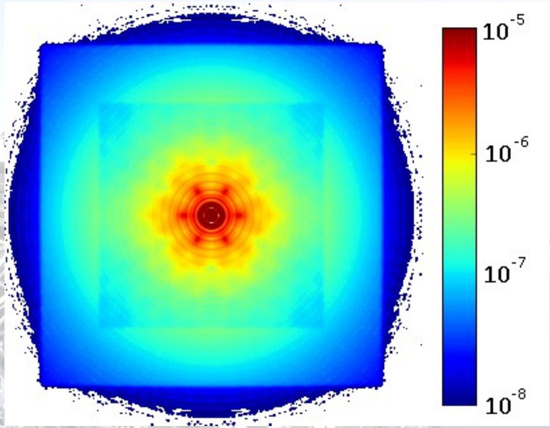
- Wide-field mode
  53” FoV with 4mas/pixel
- Zoom mode
  16” FoV with 1.5mas/pixel
- Spectroscopy mode
  R~4000 with 3” slit

Things to know about SimCADO

- SimCADO is designed to **run on a laptop**

  Limited to ~4GB RAM and ~4 cores

MAORY Ks band PSF (2.2µm)

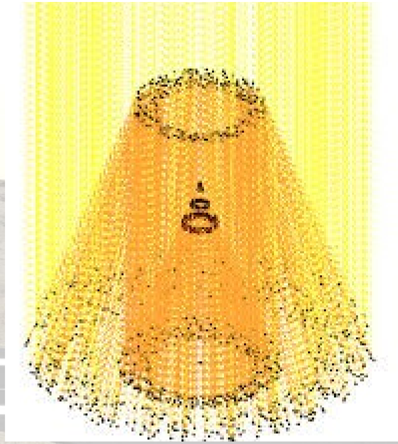


- SimCADO combines **data** supplied by **other work packages**

  E.g. MAORY, SCAO PSFs, detector layout, distortion...
  SimCADO is only as accurate as the input data given to us from sub-package simulations

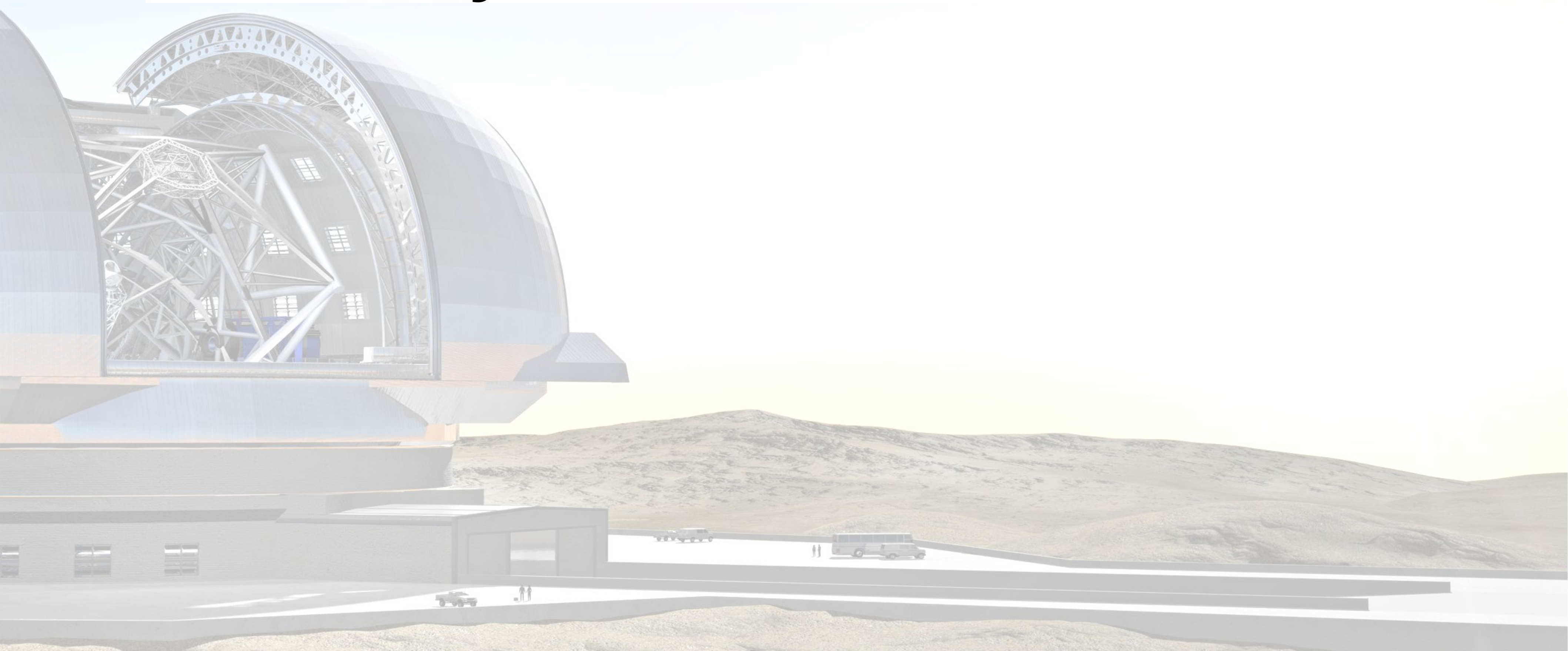
- SimCADO does **neither** ray-tracing nor physical optics

  Enough said



http://iopscience.iop.org/article/10.1088/1748-0221/9/04/C04010/pdf

Why build SimCADO?



SimCADO has 4 potential user groups

Science Team

For determining the observable **limits and feasibility** of science cases

Data flow team

For generating **mock input data** for the development of the various data flow pipelines

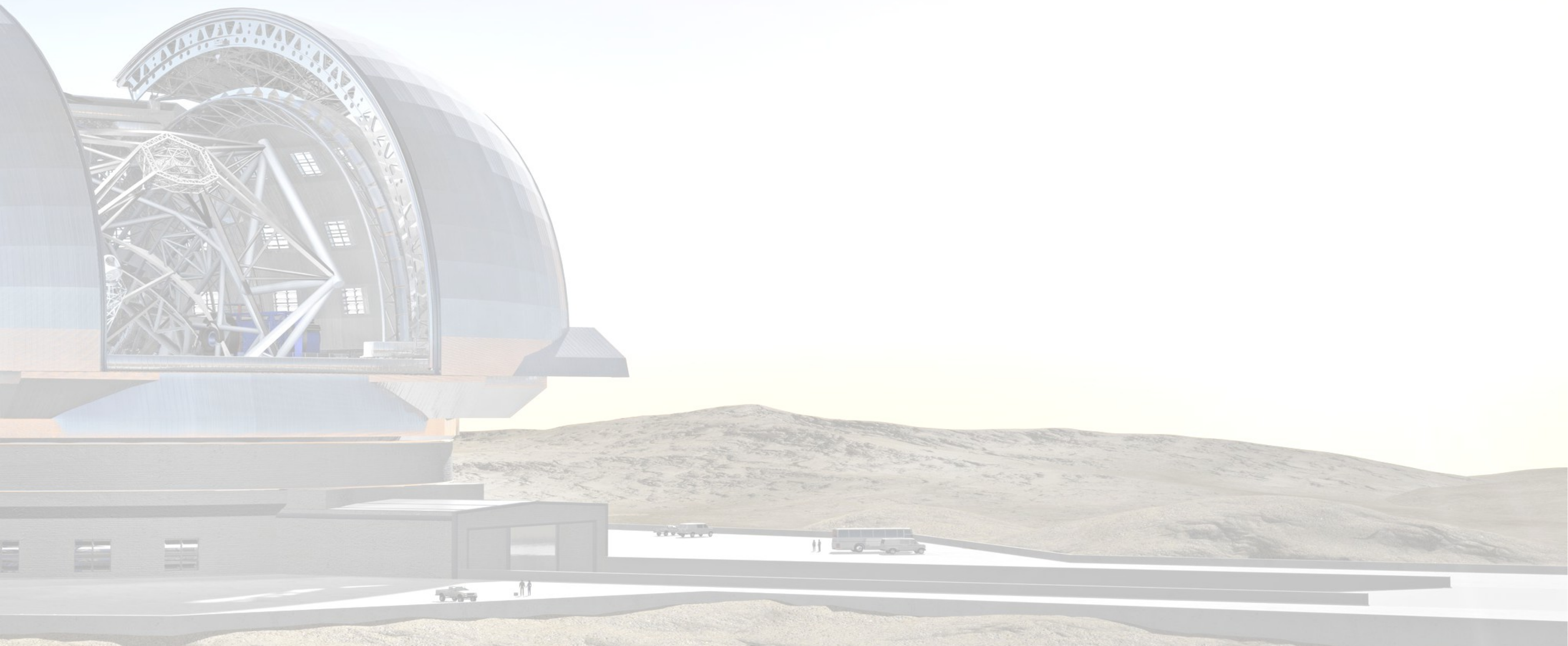
Instrument design team

For visualising the **effect of different components** on the science quality of the output images

Astronomical community

For testing out the **observability** of future science cases as well as preparing observations and/or optimising observation strategies

How does SimCADO work?



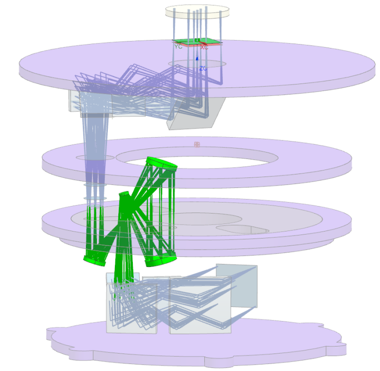
SimCADO mimics changes to the incoming source photons to produce detector-array readout files

- the atmosphere
- the E-ELT
- MICADO
- the detector plane array

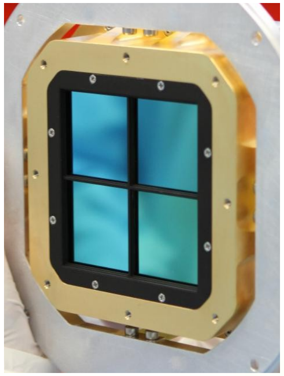
Source

Atmosphere

Telescope



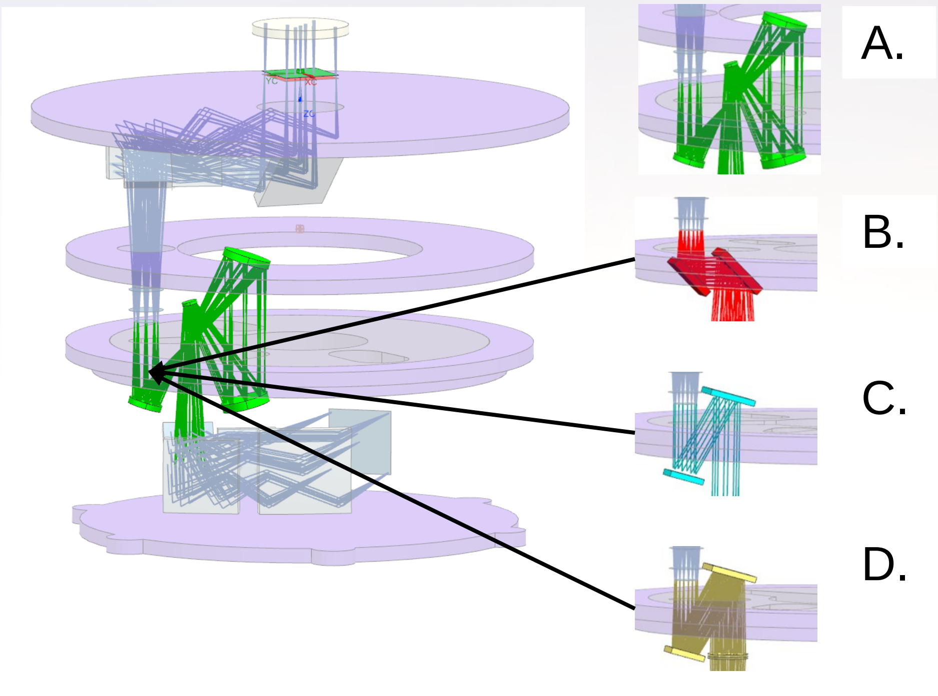
MICADO



H4RG

The MICADO optical train contains between 25 and 35 elements

- Collimator and Re-imager
- 2x Filter wheels
- Mask wheel
- ADC
- Derotator
- 3x3 H4RG detector array
- Between 9 and 20 mirrors (excl./inc. MAORY)



A. 1.5mas imager (4 fixed mirrors)

B. 4mas imager (2 flat fold mirrors)

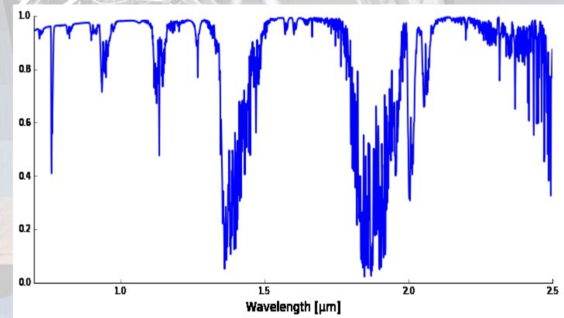
C. Cross-dispersed Spectroscopy (2 gratings)

D. Pupil imager (2 flat fold mirrors + 1 lens)

Each element affects the incoming photons differently

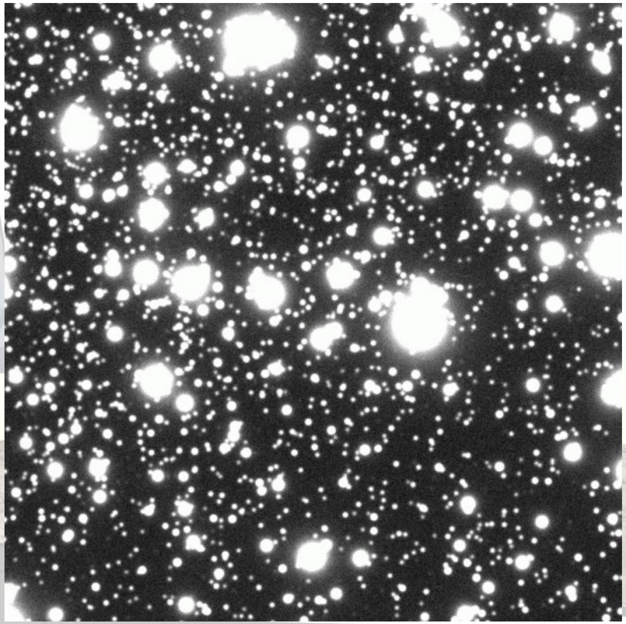
Spectral (λ)

- throughput
- atmospheric emission
- blackbody emission
- ...



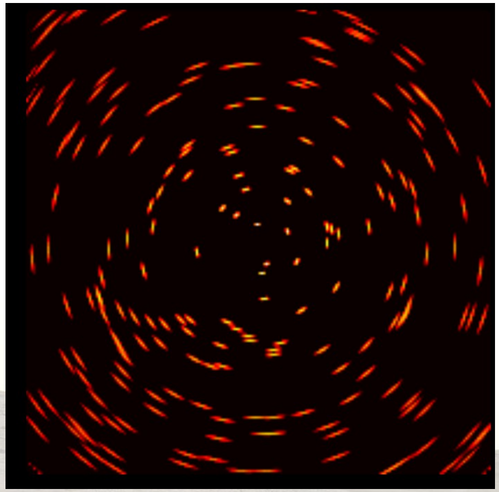
Spectrospatial (x,y,λ)

- PSFs
- atmospheric dispersion
- ...

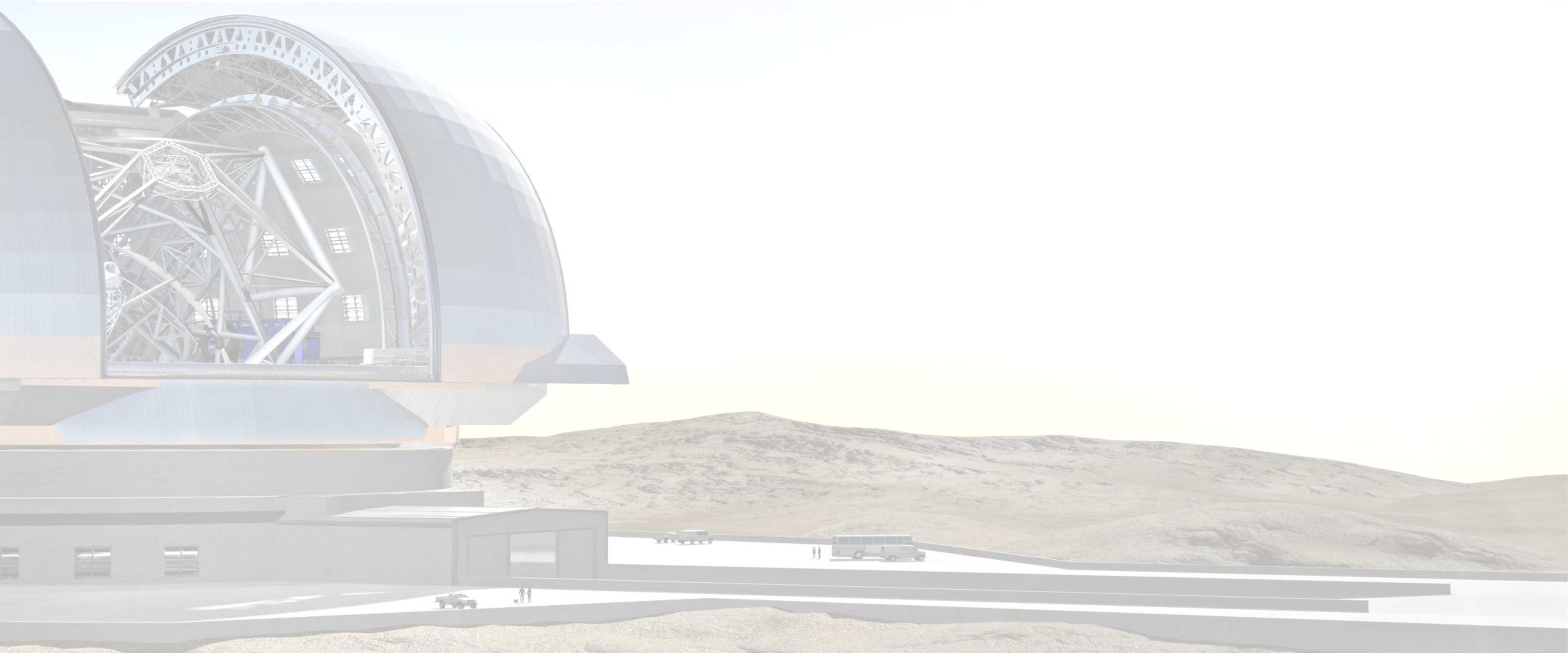


Spatial (x,y)

- translation
- rotation
- distortion
- vibration
- ...

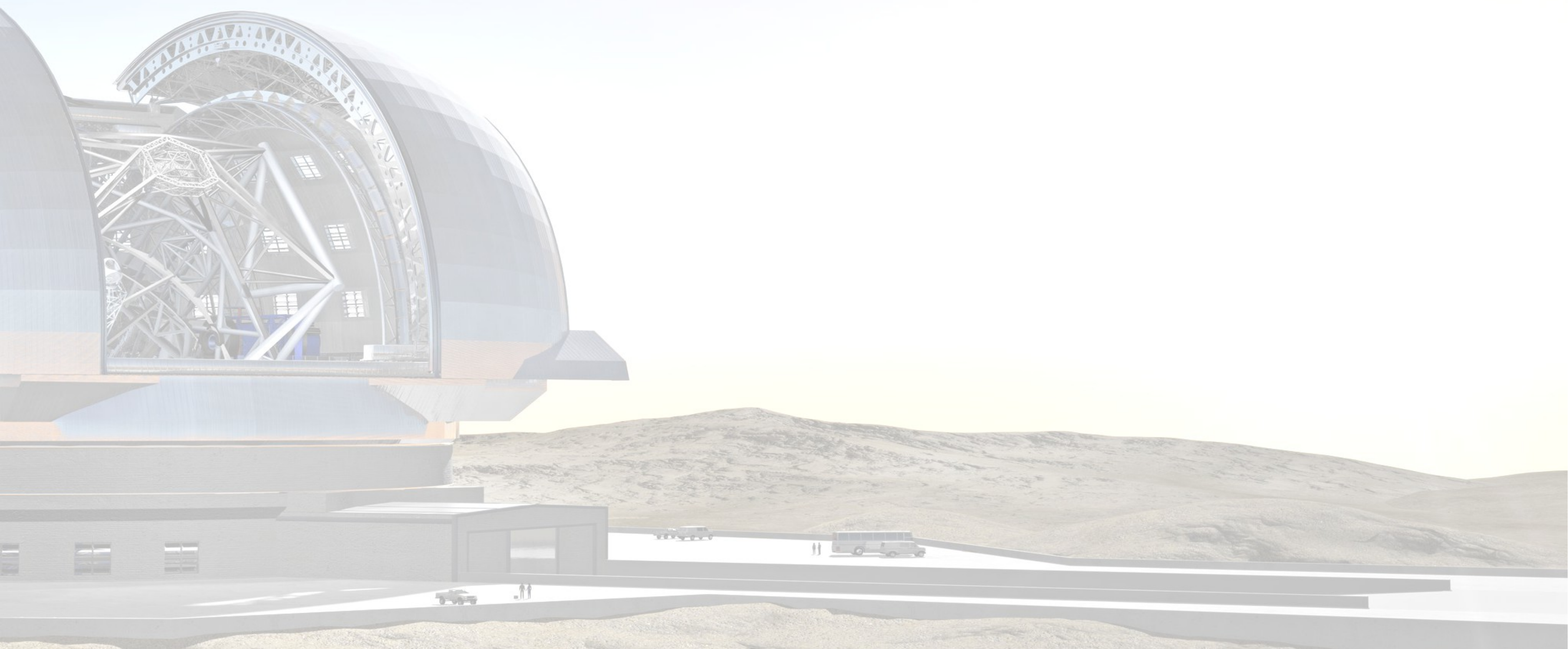


How does SimCADO simulate?



SimCADO contains 3 workhorse classes

`Source` `OpticalTrain` `Detector`



Source objects contain 2 tables

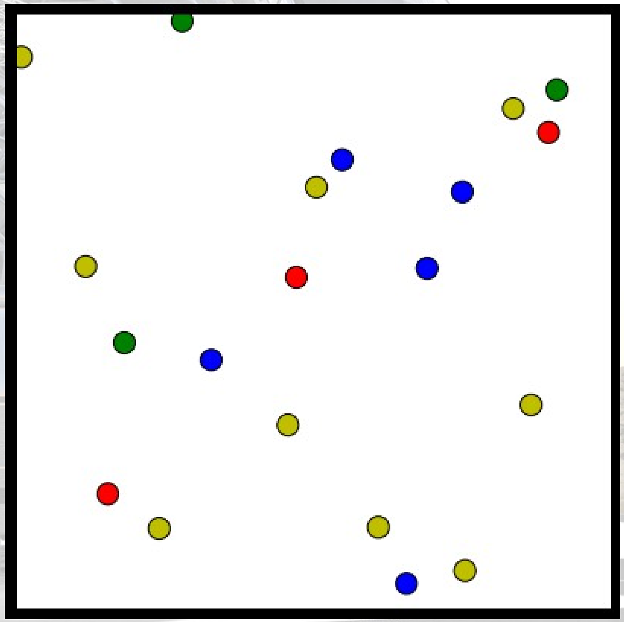
Source

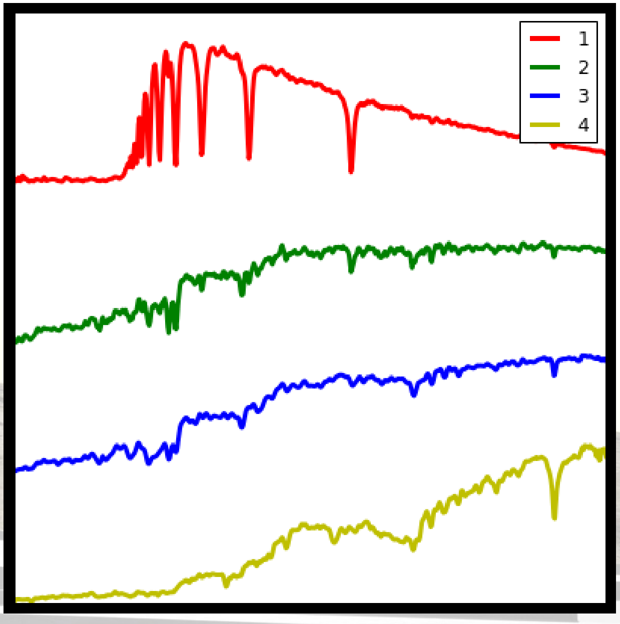
Table 1 (x,y,ref,w)

contains the positions of sources and references to their spectra

Table 2 [..λ..]

contains a list of *unique* spectra

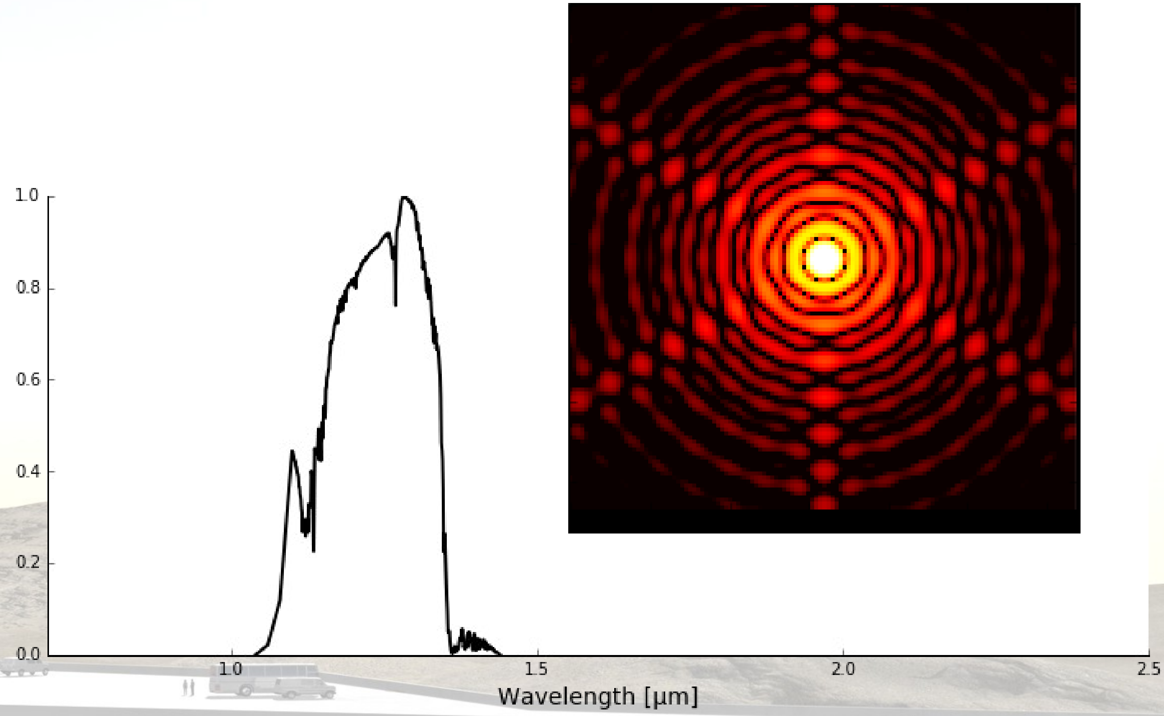




OpticalTrain holds a **collection** of **transformations** that need to be applied to the **Source**

OpticalTrain

- psfs
- transmission curves
- tracking shifts
- rotations
- distortions



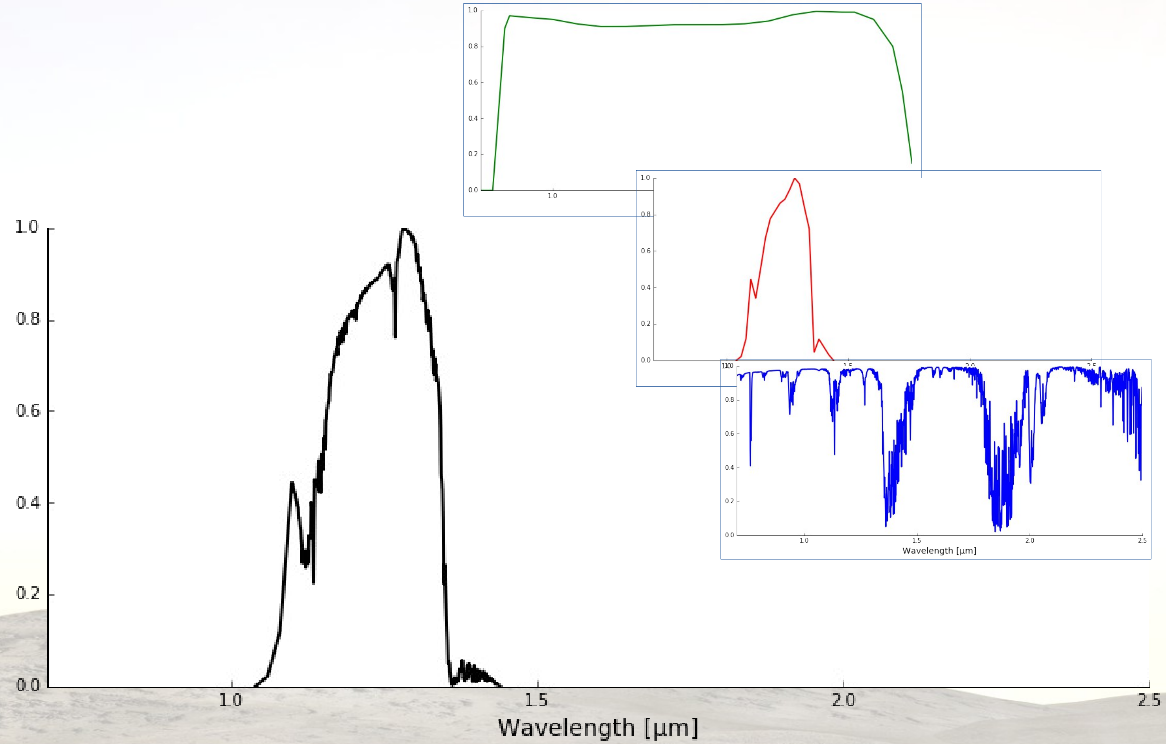
Each object in OpticalTrain represents the sum of a specific type of effects

OpticalTrain

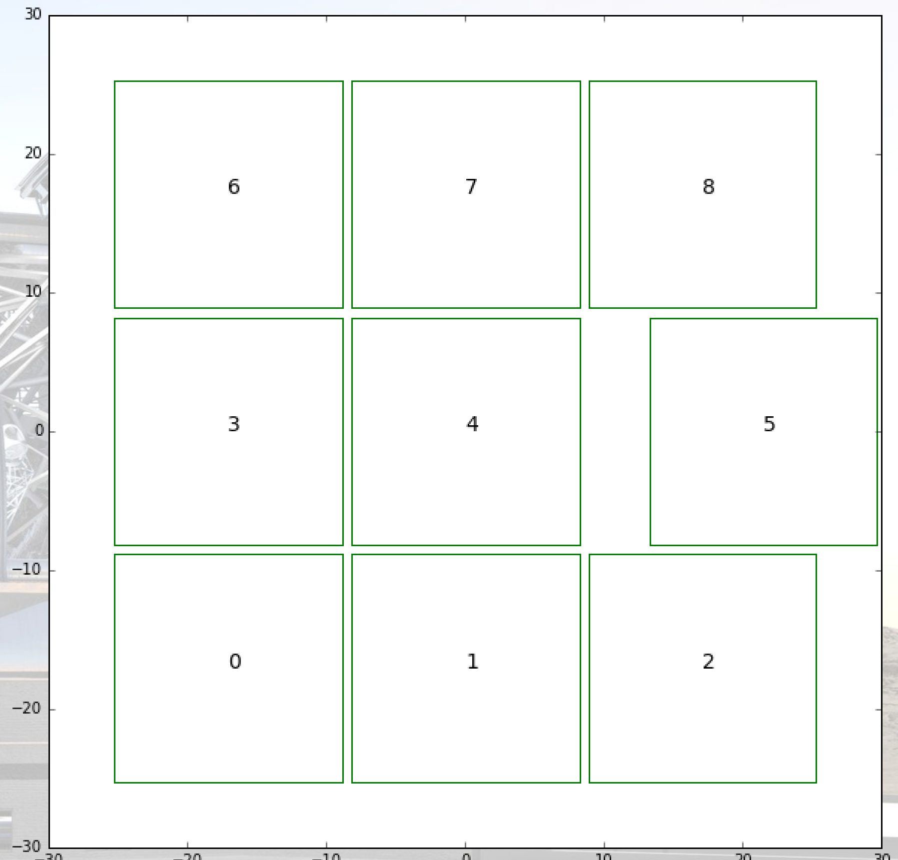
For example:

OpticalTrain.tc_source

contains the product of all relevant transmission curves from the source to the detector



Detector describes the geometry of the focal plane array and contains a list of **Chip** objects



Detector

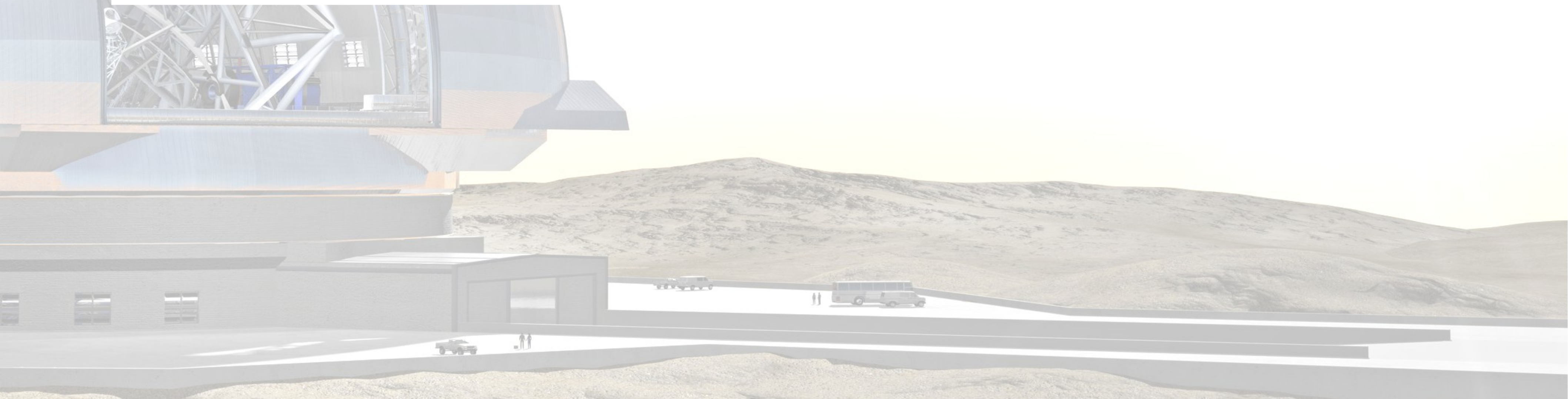
Detector contains the physical information about the focal plane array

Chip objects contain the “images” data

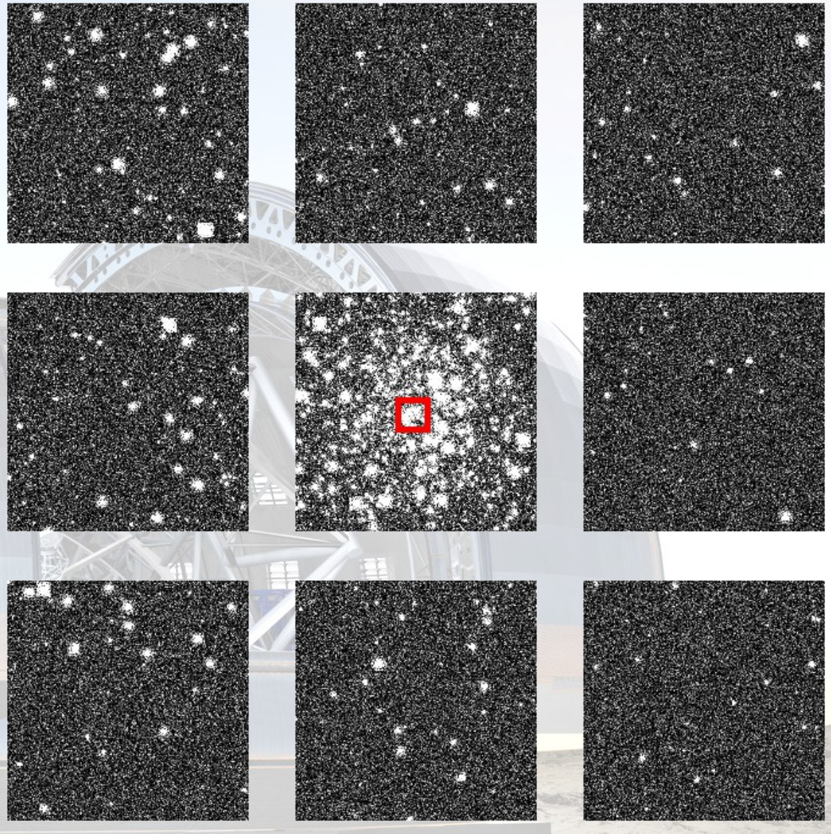
`Source.apply_optical_train(OpticalTrain)`

The main body of the simulation is executed when this method is called, namely:

- all transformations are applied to the tables (where possible)
- an image is generated
- image operations are executed (rotation, jitter, ...)
- image is placed “on” the `Detector` `Chips`



`OpticalTrain.Detector.Readout()`



- noise is added (all forms)
- the `Chips` are read out according to the desired readout scheme (e.g. Up-the-ramp, Fowler, …)
- a FITS file is created (or astropy HDUList object)

Controlling SimCADO can be done in two ways:

ASCII configuration file

Pass a "SExtractor"-style config file with the relevant parameters when using SimCADO via the CLI

```
OBS_FOV                  16     # [arcsec] side length of the field of view
OBS_EXPTIME              60     # [sec] simulated exposure time
OBS_NDIT                 60     # [#] number of exposures taken
OBS_NONDESTRUCT_TRO      1.3    # [sec] time between non-destructive readouts
OBS_REMOVE_CONST_BG      yes    # remove the minimum background value

OBS_INPUT_DIR            none   #
OBS_INPUT_NAME           none   #
OBS_FITS_EXT             0      # the FITS extension number

################################################################################
# Parameters relating to the simulation

SIM_OVERSAMPLING         1      # The factor of oversampling inside the simulation
SIM_DETECTOR_PIX_SCALE   0.004  # [arcsec] plate scale of the detector
SIM_PIXEL_THRESHOLD      1      # photons/pixel summed over the wavelength range
```

`UserCommands` object

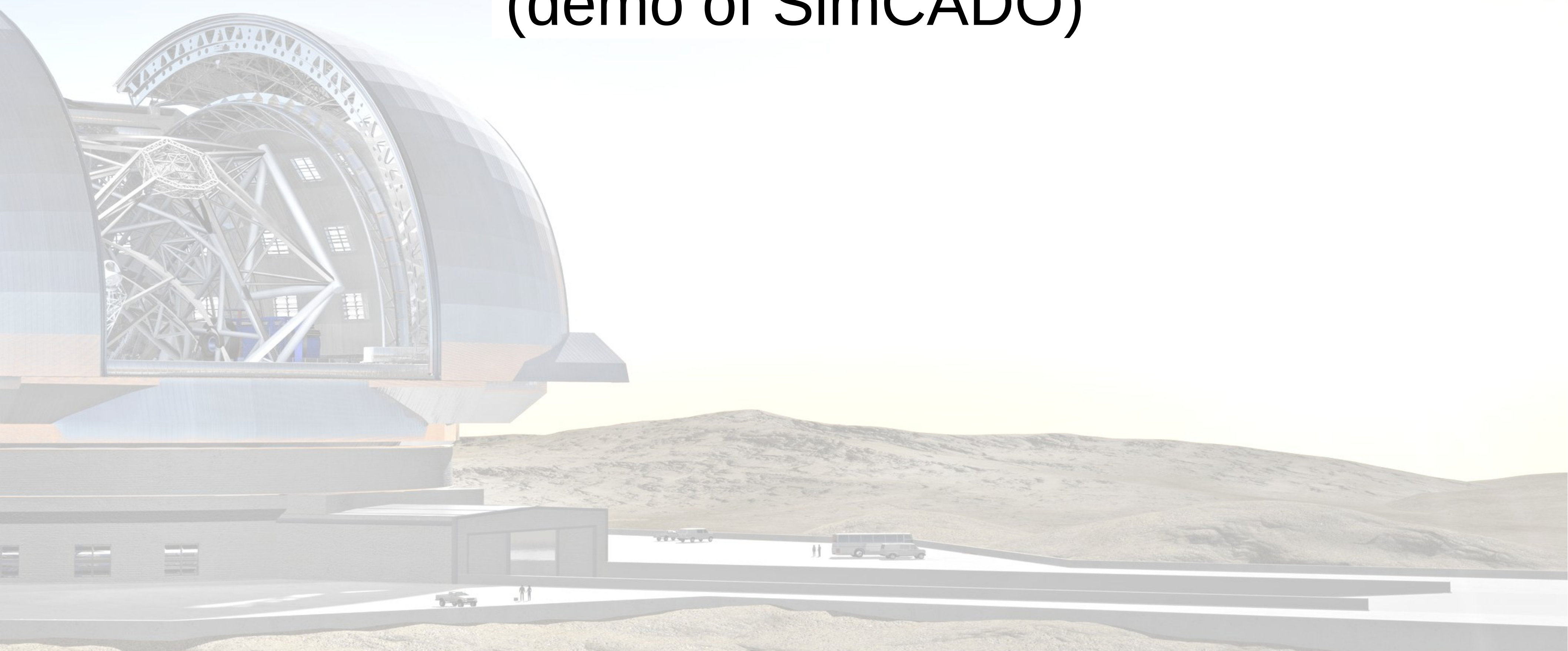
Contains all the default parameters and can be changed interactively in iPython

```
In [159]: my_cmds = simcado.UserCommands()
          my_cmds.atmo

Out[159]: {'ATMO_BG_PHOTONS': 0.0,
           'ATMO_EC': None,
           'ATMO_PRESSURE': 750.0,
           'ATMO_REL_HUMIDITY': 60.0,
           'ATMO_TC': '../data/skytable.fits',
           'ATMO_TEMPERATURE': 0.0}

In [ ]:
```

Let's play
(demo of SimCADO)

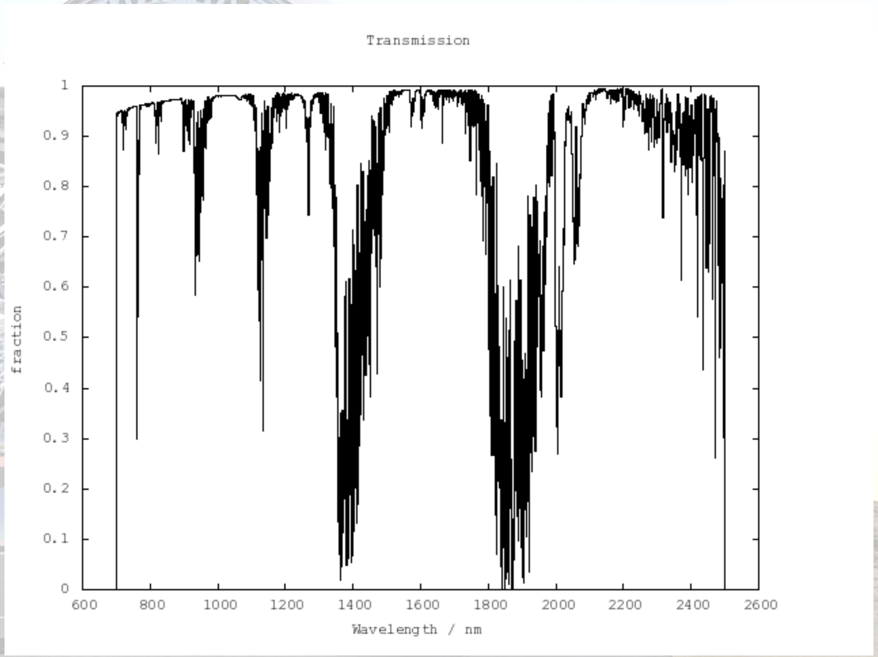


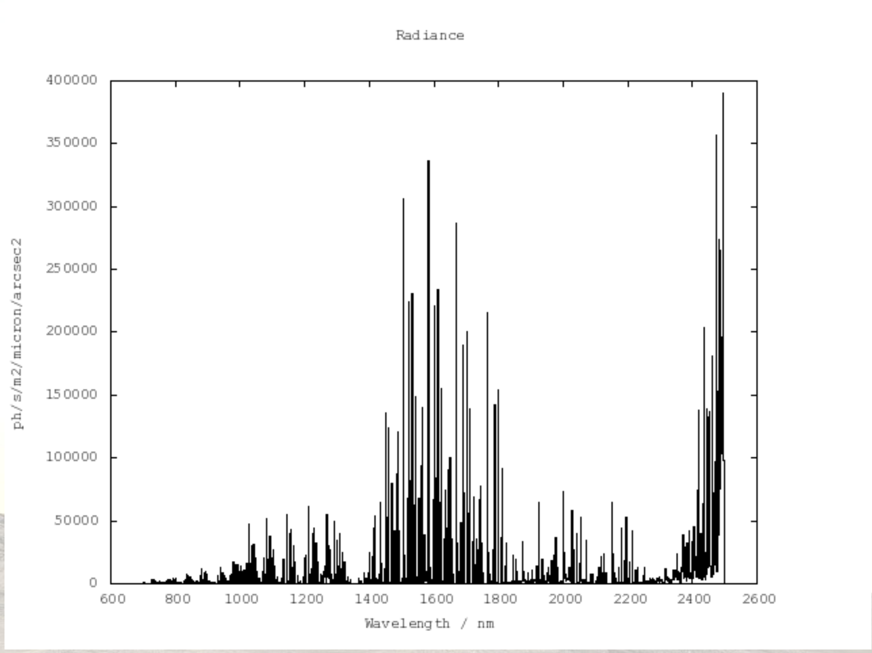
3rd party code

SkyCalc provides model atmospheric transmission and emission data

https://www.eso.org/observing/etc/bin/gen/form?INS.MODE=swspectr+INS.NAME=SKYCALC

Developed by the IAT in Innsbruck for ESO

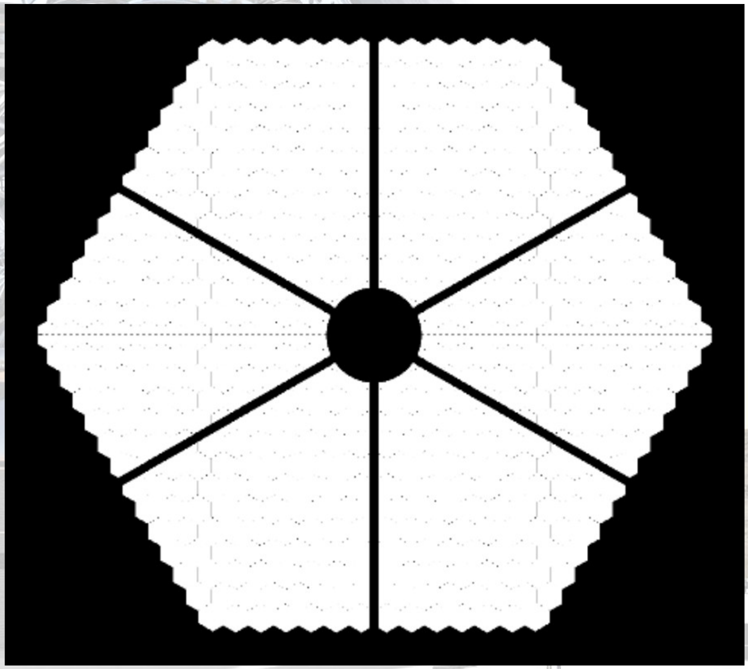


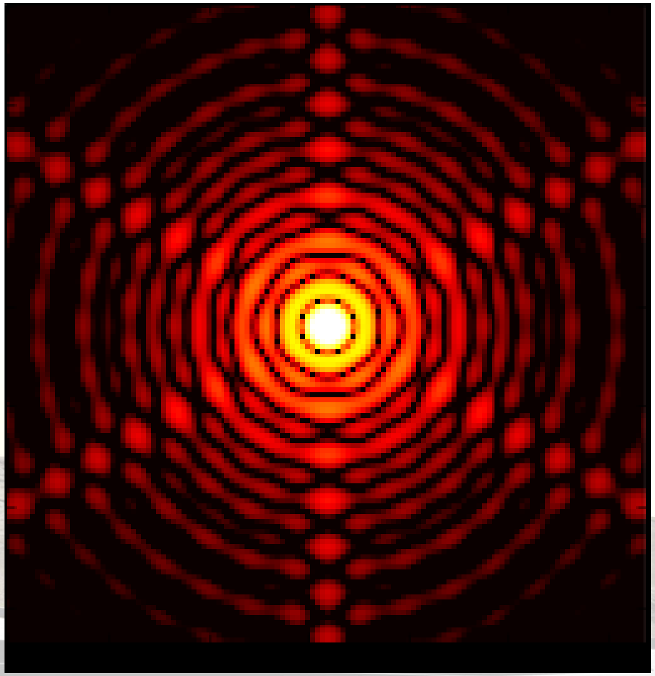


JWST POPPY generates PSFs for mirrors comprised of N hexagonal segments

https://pythonhosted.org/poppy/

- Scalable to 39m, however circular mirrors are difficult
- Generates ideal case PSFs, i.e. an *unrealistic* perfect AO system
- Installable via pip, conda, easy_install, ...



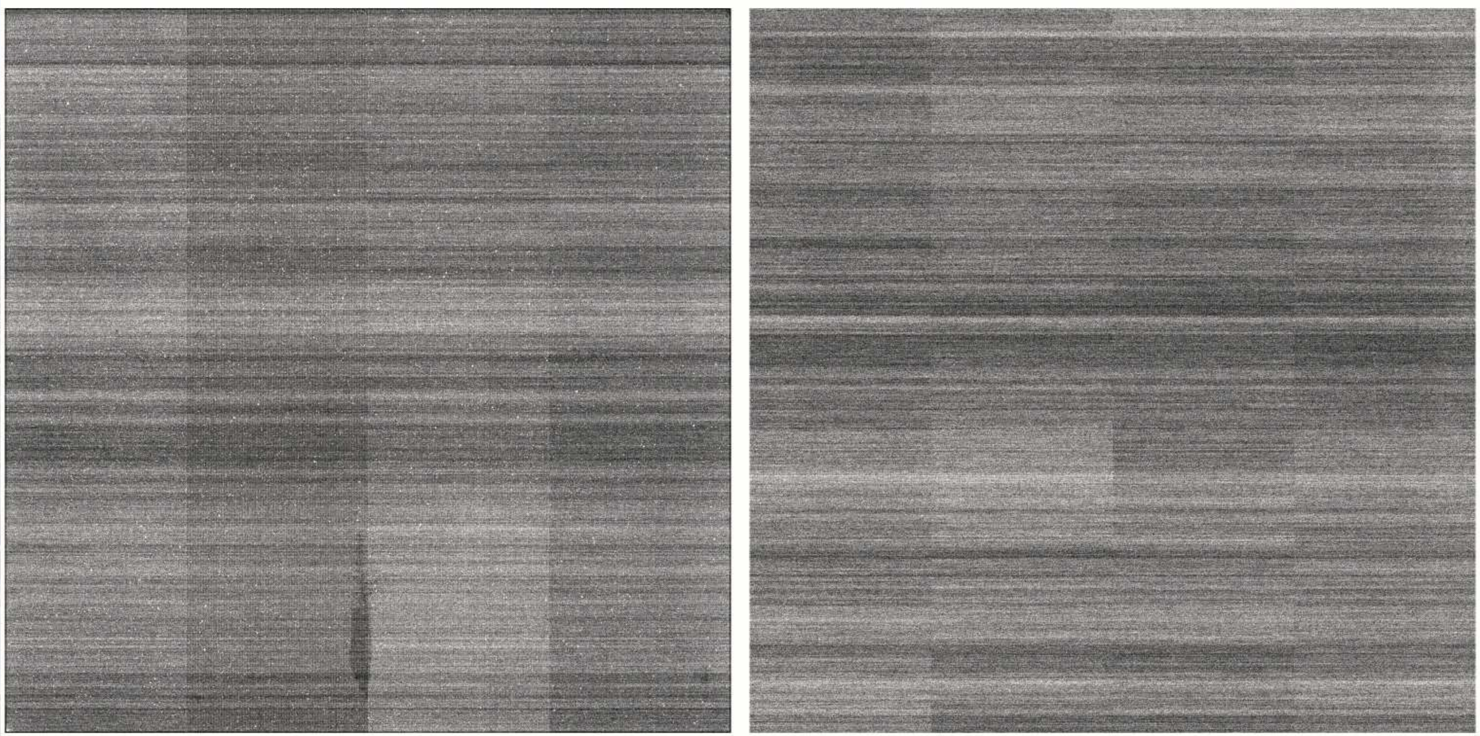


Bernhard Rauscher's HxRG Noise Generator

http://jwst.nasa.gov/publications.html

Python code which does exactly what the name suggests

Scalable for all detectors in the Hawaii RG series



(a) Real data (b) Simulated data

Rauscher, 2015 - http://adsabs.harvard.edu/abs/2015PASP..127.1144R

The main problems were related to computer memory and lack of input data

Restricted by the amount of RAM in a typical laptop (~4 to 8GB)

Memory requirements for brute force approach

4k x 4k	: 16 Mpixel
4 btye / pixel	: 64 MB
9 chips	: 576 MB
4x oversampling	: 9.2 GB
R ~ 40	: 64 GB

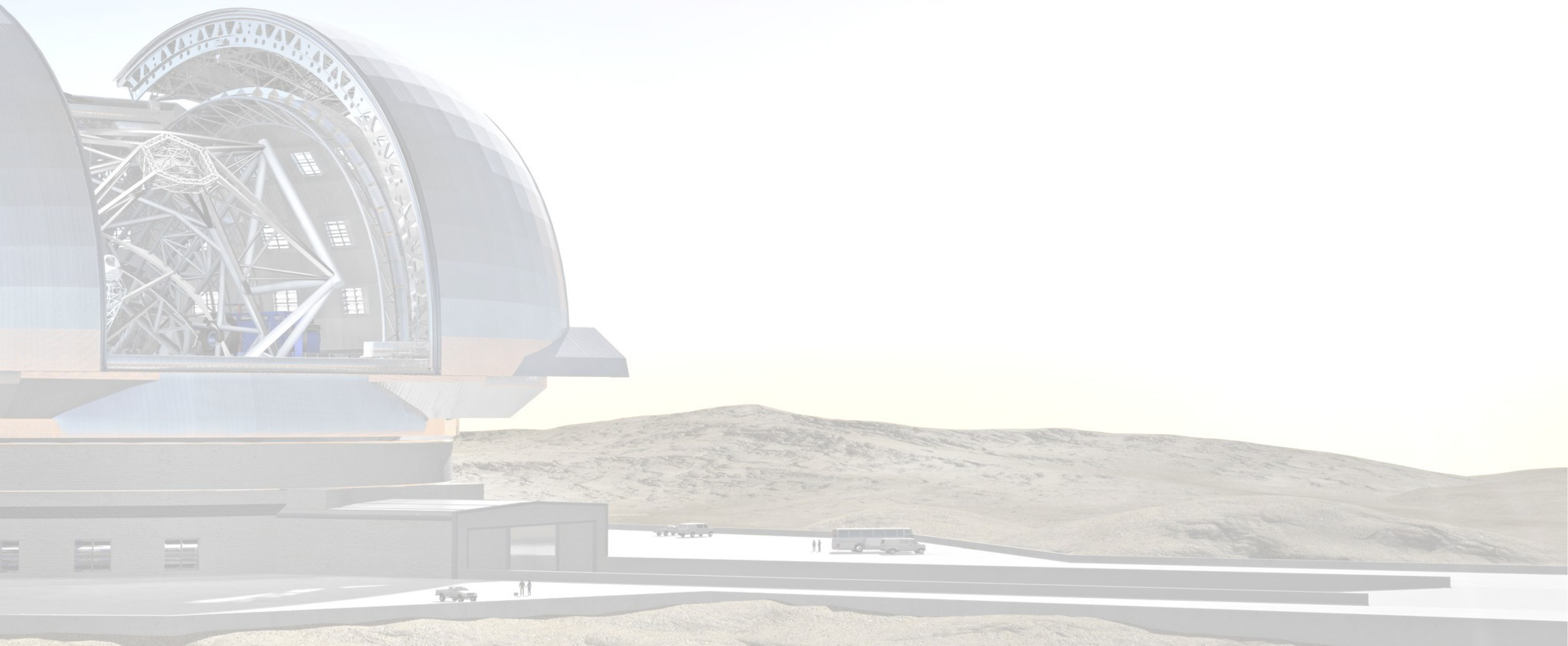
Novel solutions were found to circumvent SimCADO's thirst for RAM

Much data is not yet available. This caused us to make assumptions about various aspects:

e.g.

- Distortion map
- MAORY / SCAO PSF cubes
- ADC specs
- Detector Persistence, Linearity
- …..

Re-capping SimCADO



SimCADO

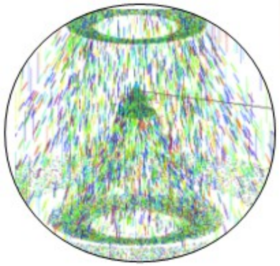
An Instrument Data Simulator for MICADO

- is a distributable python package
- combines results from other MICADO work packages
- simulates detector readouts on a laptop
- decentralises the simulation effort
- produces simulations quickly and efficiently

Discussion

How does SimCADO fit into the instrument simulator landscape?

HARMONI



SimCADO

An Instrument Data Simulator for MICADO

4MOST

MOONS

Multi Object Optical and Near-infrared Spectrograph

METIS

Mid-infrared E-ELT Imager and Spectrograph

Questions for the Audience

- How do the users interact with your simulators?
- What kinds of infrastructure are needed to run the simulations?
- How will the simulator change with time as the instruments become more developed?
- Is the simulator mainly used by the science team, or does the design team also play with it?
- Is anyone dealing with variable IR backgrounds?
- What testing structures do you use?
- How did you solve the memory problems?